UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF NEW YORK

MOOG INC.,

Plaintiff,

v. Case No. 22-cv-00187

SKYRYSE, INC., ROBERT ALIN PILKINGTON, MISOOK KIM, and DOES NOS. 1-50,

Defendants.

DECLARATION OF KEVIN M. CROZIER

KEVIN M. CROZIER, under penalty of perjury and pursuant to 28 U.S.C. § 1746, declares the following to be true and correct:

I. Background

1. My name is Kevin M. Crozier. I provide this declaration in support of Moog Inc.'s ("Moog") Opposition to defendant Skyryse, Inc.'s Motion to Enter Source Code Protocol. I am over the age of 18 years. I have personal knowledge of the matters set forth herein and if called as a witness, I could and would competently testify as to all facts set forth herein.

2. I have been retained by Sheppard, Mullin, Richter & Hampton LLP, counsel for Moog Inc. ("Moog" or "Company") to, among other things, analyze and opine on defendant Skyryse, Inc.'s ("Skyryse") flight control source code, software, and other software-related documents to assist Moog in determining to what extent Skyryse has misappropriated, copied, or otherwise used Moog's trade secrets.

3. I have a Bachelor of Science degree in Electrical Engineering with a Minor in Computer Science from Rensselaer Polytechnic Institute and Master of Science degree in Electronical Engineering from the University of Illinois. I am a Federal Aviation Administration (FAA) consultant Designated Engineering Representative (DER) for safety critical avionics software with more than two decades of hands-on industry experience in software development. I have specialized technical experience in embedded systems and software development, compiler design and implementation, processor architecture, multi-core network processors (NPUs), software simulation, and network security protocols.

4. From 1999-2000, I worked for Hewlett Packard (HP) developing compiler technologies for 64-bit microprocessors using C++. At HP, I was granted 2 patents for floating point code optimizations that I developed and implemented.

5. From 2000-2005, I worked for Teja Technologies where I designed and implemented a C--like language to express state machines used to program multicore NPUs. I was the technical team leader for the Teja toolchain port to AMCC NPU from requirement definition through implementation. I was also a lead software architect on customer projects, with tasks that included defining software architectures, creating IXP processor microcode, writing embedded C for the VxWorks real-time operating system, analyzing performance, and performing functional testing.

6. From 2005 to 2013, I worked for Cavium Networks (now Marvell) as the lead customer applications architect for ARM-based networking, set top box, and wireless display system-on-chip (SOC) products, the OCTEON multicore MIPS-based processor, the NITROX security co-processor, and the ECONA multicore ARM-based processor. I architected numerous designs for strategic customers, including integrated service routers, enterprise IPSec security

platforms, network service platforms, SSL load balancers, application load balancers, storage networking modules, enterprise wireless access points, enterprise wireless controllers, home router-gateways, home wireless access points, satellite networking equipment, consumer NAS devices, and network test equipment. I also developed device drivers and embedded networking applications in C, C++, and assembly for platforms based on Linux v2.6 as well as bare metal MIPS, and u-boot.

7. From 2013 to 2016, I worked for BendixKing by Honeywell as the director of software engineering and chief software architect for a next generation of avionics and flight displays for general aviation aircraft. I co-led a successful hardware and software prototyping effort to prove software and hardware platform feasibility for GPS/FMS product and communication radio product. I developed a streamlined system and software development process based on Agile methodologies to optimize the efficiency of a small co-located development team for a fully integrated and modular primary and multi-function flight display. I helped define the system requirements and software architecture and led software design, development and review in C, C++, and assembly. I successfully developed, implemented, and refined a DO-178C software process and infrastructure from the ground up, including requirements capture and review; code development and testing; and quality assurance.

8. Since 2016, I have worked as a FAA consultant DER ensuring that engineering data for aircraft software systems complies with the FAA's airworthiness standards. My FAA designation allows me to approve software engineering data on small and large airplanes (part 23 and 25) and helicopters (part 27 and 29) up to the highest level of level of safety critically (DAL A).

9. I am also a commercial pilot and certified flight instructor with over 2800 hours of total flight time and over 650 hours of dual instruction given.

10. Pursuant to the procedures set forth in the Protective Order entered in this case (ECF 89) and the Inspection Protocol (ECF 96-02), I have been granted access to certain electronic devices and data turned over to iDS by all parties in the case through inspection laptops and iDS' remote virtual machine software.

II. A 100 Page Limit on Source Code Printing is Insufficient

11. I have reviewed Skyryse's Motion to Enter Source Code Protocol and Skyryse's proposed second source code protocol attached as Exhibit "A" thereto. (ECF 213).

12. During my inspection of the Moog and Skyryse devices and data turned over to iDS, I have noted several examples of Skyryse's direct copying of Moog software and documents. These examples are discussed in further detail in my August 3, 2022 declaration at Paragraphs 12-17. (ECF 210-001).

13. I understand that Skyryse is proposing a 100 page printing limit on source code. Based on my experience, as well as my review of certain Skyryse electronic devices produced to iDS to date, a 100 page limit would not be appropriate for this case. My understanding is that the volume of files coped by the defendants from Moog is over a million, and that there are tens of thousands of source code files at issue. For example, just the 24 identical or near-identical Skyryse source code files that I discuss in Paragraphs 12-14 of my August 3 declaration constitute approximately 20,000 lines of code, or approximately 300 printed pages. This is just one example of many that I intend to include in a future expert report or declaration regarding Skyryse's use of Moog's source code and other proprietary information.

III. A Direct Comparison of Moog and Skyryse Source Code is Required

14. I understand that under Skyryse's proposed second source code review protocol, a reviewer would be required to travel to Skyryse's counsel's offices in-person to review Skyryse's code on a computer. I further understand that, under Section 4.3(c) of the proposed protocol, I would only be permitted to bring Moog's source code on paper or on a separate computer. I understand that Skyryse's proposal does not permit the comparison of Moog and Skyryse source code on the same computer.

15. Based on my experience, and my involvement and work on this case, a direct comparison of Moog and Skyryse source code on the same computer is required. Both Moog and Skyryse code needs to be available on the same computer so that file comparison tools can be used. Comparison tools such as UltraCompare allow the comparison of the contents of files with a host of features that make spotting differences between files much more efficient and accurate. This tool can be configured to ignore white space or other characters or structures, highlight the file differences, compare entire directories of files, and produce compare result reports. The importance of direct comparison of source code is exemplified by the direct comparison I have already performed on Moog's and Skyryse's respective source code to identify the examples of direct copying by Skyryse in my August 3 declaration.

16. Without the ability to use source comparison tools, I and Moog's other experts would be forced to perform manual comparisons, which will be very slow and error-prone since we would be required to look between two different computers (or between printouts of code and a computer) and then manually identify the similarities and differences. A comparison of code using comparison tools is much more accurate and efficient, especially given the volume of source code and proprietary files in this case.

IV. Software Design Documents, Software Requirements Documents, and Software Checklists Are Highly Critical

17. I understand that Skyryse's proposed second source code review protocol would only apply to source code documents, and not to other software-related documents as such as software design documents, software requirements documents, and software checklists. However, based on my experience, these types of non-code software documents are extremely valuable to avionics companies, and are highly critical to any substantive analysis of flight control software trade secret copying and use.

18. The development of safety critical software used in aircraft must follow a well-defined software development process which is tailored to meet the objectives of software certification as defined in DO-178C. Compliance with the objectives of DO-178C is the primary means for meeting airworthiness requirements and obtaining approval of the FAA for the software used in civil aviation products including automatic flight control systems (autopilots), flight management systems, and even passenger entertainment systems. DO-178C prescribes numerous objectives that must be met before software will be approved to fly on an aircraft.

19. The first step in meeting these objectives is to develop plans which document the software development process for the specific project. There are typically a number of software plans, software standards, and checklists which are created to define this process. These plans are followed by all of the software engineers during the course of the project. The associated checklists are used to confirm that the document, code, or test developed by the engineer will meet the appropriate objectives of DO-178C as required for certification.

20. Based on my experience and background, an established company will make templates of these documents and checklists and then reuse them (with some tailoring) for each

software project they undertake. These plans and standards are typically regarded as company IP as there is significant effort that has gone into them for their initial development and updating as the company's software process has evolved.

21. When a new avionics company starts to develop its own process, it generally must either purchase these checklists and plan templates from another company or develop them from scratch. Generally, the cost is significant for either route.

I declare that the foregoing is true and correct under penalty of perjury under the laws of the United States of America.

Dated: August 17, 2022

Kevin M. Crozier